**Exam #2 (100 points) -- CS 4080** *: Ocean Lu*

1. *This is a close-book test. Type the answers and save as a pdf file to submit on blackboard.*
2. *If you feel a question contains an error or ambiguity please say so in your answer and provide the answer based on your best judgment.*
3. *60 minutes total. Good luck!*

Problem 1: (15 points) Short answer. (“key” words only, no need complete sentence.)

1. What are the two kinds of abstractions in programming languages?

**The two kinds of abstractions in programming languages are process abstractions and data abstractions.**

1. From where are the C++ objects allocated?  
   **C++ objects are allocated on stack and heap.**
2. Why are ***destructors*** rarely used in Java but essential in C++?

**Destructors are rarely used in Java, but essential in C++ because Java has its own garbage collecting program.**

1. What is the key difference between Java’s ***for each loop*** and regular Java ***for loop***?

**The key difference between Java’s for each loop and Java’s for loop is that the for each loop is logically controlled while the for loop is counter controlled.**

1. What is the advantage and disadvantage of introducing ***break*** statements in a language?  
   **The disadvantage of introducing a break statement in a language is bad reliability and readability.**

Problem 2: (10 points) **True or False.**

1. Class A and Class B are two independent classes, i.e. no one inherits from the other. A has a method ***void fun (int item)*** and B also has the method ***void fun (int item)***. These two fun methods are overriding method.

**True**

1. Class A has two methods ***void fun (int item)*** and ***void fun (int item, double cost)***. These two fun methods are overloading methods.

**True**

1. A lambda function is a function that could take another function as parameter.

**True**

1. When a class extends or inherits from more than one parent classes, this is called \_\_\_\_\_\_\_\_.  
   **Inheritance**

Problem 3: (12 points) Short answer. (“key” words only, no need complete sentence.

1. C++ has virtual function and pure virtual function. What is the main difference between these two?  
   **The main difference between virtual and pure virtual functions is that virtual functions can be called through polymorphic variables and dynamically bound to messages. A pure virtual function has no definition at all.**
2. Java doesn’t have virtual function or pure virtual function. How does Java implement the feature of C++’s pure virtual function?  
   **Java implements C++’s pure virtual function through using abstract methods.**
3. How C++ implements a generic ADT?  
   **C++ implements a generic ADT through encapsulation devices. It creates instances of the class that share a single copy of the class data members.**

Problem 4: (15 points) Operator overloading.

1. Why some languages introduce operator overloading? (To answer this question, first list a criterion that this feature is good for (i.e. advantage), then use a simple example with pseudo codes to illustrate this advantage of operator overloading.)

**Advantage: allows readability and flexibility of calling similar methods with different types of data  
Pseudo code for advantage:  
void fun(int a) //maybe does something fun with integer a  
void fun() // maybe does something fun on default if we don’t have integer a**

1. For each of the languages below, please circle the ones that support user-defined operator overloading.
2. Java **(2) C++ (3) C#**
3. If a language doesn’t support operator overloading, could you name a reason behind it?  
   **Languages that don’t support operator overloading is probably due to the complication with parameter coercions, which complicate the disambiguation process.**

Problem 5: (24 points)

Given the following C++-like description of an inheritance hierarchy.

class Animal:

//constructors and destructors (if needed) are properly defined

void cry (); //an abstract method

void fur() { … } //a proper method with details defined

double weight( … ) { … } //a proper method with details defined

class Dog: inherits from class Animal

//constructors etc. properly defined

void cry() {…. } //a proper method with details defined.

void fur() {… } //a proper method with details defined.

Animal myPets = new Animal [2];

myPets[0] = new Animal();

myPets[1] = new Dog ();

1. A method call, *myPets[0].cry()* , under static binding, which method will it bind to? Under dynamic binding, which method will it bind to? (note: indicate “error” if such a call will result an error. Same for the following questions.)

**Static binding: the method it binds to is class Animal void cry()  
Dynamic binding: the method it binds to is class Animal void cry()**

1. A method call, *myPets[1].cry()* , under static binding, which method will it bind to? Under dynamic binding, which method will it bind to?

**Static binding: the method it binds to is class Animal void cry()  
Dynamic binding: the method it binds to is class Dog void cry()**

1. A method call, *myPets[1].fur()* , under static binding, which method will it bind to? Under dynamic binding, which method will it bind to?

**Static binding: the method it binds to is class Animal void fur()  
Dynamic binding: the method it binds to is class Dog void fur()**

1. A method call, *myPets[1].weight()* , under static binding, which method will it bind to? Under dynamic binding, which method will it bind to?  
   **Static binding: the method it binds to is class Animal void weight()  
   Dynamic binding: the method it binds to is class Animal void weight()**
2. Does C++ support static binding? If yes, how does it support?

**Yes, C++ supports static binding by using function overloading and overriding.**

1. Does C++ support dynamic binding? If yes, how does it support?  
   **Yes, C++ supports dynamic binding by using virtual functions.**
2. Does Java support static binding? If yes, how does it support?  
   **Yes, Java supports static binding by using function overloading and overriding.**
3. Does Java support ~~static binding and/or~~ dynamic binding? If yes, how does it support?  
   **No, Java does not support dynamic binding.**

Problem 6: (24 points)

1. Given the following piece of C++ or Java-like codes, assuming all parameters are pass by value, what will be the values of first, second, and result after function call?

void fun (int par1, int par2, int par3) {

int temp = par1;

par1 = par2;

par2 = temp;

par3 = par1 + par2;

return;

}

int first = 5, second = 7, result;

fun (first, second, result); //now display first, second, result

**First: 5**

**Second: 7**

**Result: null**

1. Repeat (a) now assuming all parameters are passed by reference.  
   **First: 7**

**Second: 5  
Result: 12**

1. Now given C++ code, what will be the values of first, second, and result after function call?

void fun (int & par1, int par2, int & par3) { //C++

int temp = par1;

par1 = par2;

par2 = temp;

par3 = par1 + par2;

return;

}

int first = 5, second = 7, result;

fun (first, second, result); //now display first, second, result

**First: 7  
Second: 5  
Result: 12**

1. Now given the C# code, for each parameter identify the modes (i.e. in, out, …) as well as the implementation scheme (i.e. pass by value, by reference, …)

void fun (int par1, **ref** int par2, **out** int par3) { //C#

int temp = par1;

par1 = par2;

par2 = temp;

par3 = par1 + par2;

return;

}

|  |  |  |  |
| --- | --- | --- | --- |
|  | par1 | par2 | par3 |
| Mode | **In** | **In-out** | **Out** |
| Implementation scheme | **Pass-by-value** | **Pass-by-reference** | **Pass-by-result** |